Rangkuman Pengenalan PBO & Inheritance, Encapsulation dan Polymorphism

---

Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang menitikberatkan pada penggunaan objek sebagai elemen dasar dalam pembuatan program. Objek dalam OOP adalah entitas yang memiliki atribut, perilaku, dan sering kali juga memiliki keadaan (state). OOP ditemukan pada tahun 1960, berasal dari pengembangan dari paradigma pemrograman terstruktur yang menggunakan bahasa C dan Pascal. Dengan menggunakan OOP, kompleksitas dalam menyelesaikan berbagai masalah program dapat diatasi dengan lebih efisien. Paradigma ini memungkinkan penulisan program yang awalnya sulit untuk ditangani dengan lebih mudah.

Pemrograman Berorientasi Objek (PBO) menggabungkan fungsi dan data dalam kelas-kelas atau objek-objek. PBO memiliki tiga ciri utama, yaitu Encapsulation, Inheritance, dan Polymorphism. Struktur program PBO terdiri dari kelas dan objek, yang memungkinkan penggunaan kode program yang sangat reusable dan memungkinkan penghematan space memori karena objek dan kelas dapat digunakan berkali-kali.

Pemrograman Terstruktur, di sisi lain, memecah program dalam fungsi dan data dengan tiga ciri utama: Sequence, Selection, dan Repetition. Struktur program terstruktur cenderung lebih rumit karena mengandalkan urutan proses dan fungsi-fungsi. Reusabilitas kode program kurang dalam pemrograman terstruktur karena kurangnya konsep kelas dan objek yang dapat digunakan kembali secara luas.

Class dalam pemrograman adalah cetak biru atau blueprint yang digunakan untuk membuat objek dengan kerangka dasar yang telah ditentukan. Analogi dengan kendaraan menjelaskan bahwa class mendefinisikan ciri-ciri (atribut) dan perilaku (metode) yang dimiliki oleh objek. Seperti kendaraan yang memiliki merk, mesin, dan ban, objek yang dibuat dari class memiliki atribut yang sesuai dengan ciri-ciri yang telah ditetapkan. Selain itu, objek juga dapat melakukan tindakan yang didefinisikan dalam class, seperti menghidupkan atau mematikan mesin. Dengan menggunakan class, kita dapat membuat banyak objek yang memiliki karakteristik dan perilaku yang sama, sehingga memungkinkan untuk membuat program yang lebih terstruktur dan reusable.

```
// Deklarasi Class
public class Kendaraan{
    // Ini Adalah Class, semua konstruktor, variabel, method berada disini
}
```

Dalam pemrograman berorientasi objek, objek memiliki dua karakteristik utama: atribut (state) dan tingkah laku (behaviour). Atribut adalah status dari objek yang mencerminkan karakteristik atau data yang dimilikinya, sedangkan tingkah laku adalah perilaku atau aksi yang dapat dilakukan oleh objek. Sebagai contoh, objek sepeda memiliki atribut seperti pedal, roda, jeruji, dan warna, serta tingkah laku seperti peningkatan kecepatan, penurunan kecepatan, dan perpindahan gigi.

Dalam pengembangan perangkat lunak berorientasi objek, atribut dari objek direpresentasikan sebagai variabel yang menyimpan state-nya, sedangkan tingkah laku direpresentasikan sebagai metode atau fungsi/prosedur yang menentukan perilaku atau aksi yang dapat dilakukan oleh objek tersebut. Dengan demikian, pemrograman berorientasi objek memungkinkan untuk mengorganisasi dan mengelola kode program dengan lebih terstruktur, dimana objek-objek dapat mempertahankan keadaan mereka sendiri (state) dan melaksanakan aksi-aksi sesuai dengan tingkah laku yang telah ditetapkan.

```
public class Kendaraan{

    // Konstruktor Dengan Parameter
    public Kendaraan(String nama){
        System.out.println("Nama Kendaraannya Adalah "+ nama);
    }

    public static void main(String[]args){

        // Perintah untuk membuat objek jenis
        Kendaraan jenis = new Kendaraan("Pesawat Terbang");

    }

}

// Output = Nama Kendaraannya Adalah Pesawat Terbang
```

- Inheritance

konsep dalam pemrograman berorientasi objek di mana class baru dapat dibuat dari class yang sudah ada, dengan menambahkan atau memodifikasi fungsionalitasnya. Terdapat dua jenis inheritance: tunggal (single) dan ganda (multiple). Java hanya mendukung pewarisan tunggal karena kompleksitasnya. Dalam inheritance, constructor tidak diwariskan secara otomatis ke subclass, kecuali jika menggunakan perintah super.

- Encapsulation

Cara untuk melindungi property (atribut) / method tertentu dari sebuah kelas agar tidak sembarangan diakses dan dimodifikasi oleh suatu bagian program.

- Accesors

Cara untuk melindungi data yaitu dengan menggunakan access modifiers (hak akses). Ada 4 hak akses yang tersedia, yaitu default, public, protected, private

No	Modifier	Pada class dan interface	Pada method dan variabel
1	Default (tidak ada modifier)	Dapat diakses oleh yang sepaket	Diwarisi oleh subkelas dipaket yang sama, dapat diakses oleh method-method yang sepaket
2	Public	Dapat diakses dimanapun	Diwarisi oleh subkelasnya, dapat diakses dimanapun
3	Protected	Tidak bisa diterapkan	Diwarisi oleh subkelasnya, dapat diakses oleh method-method yang sepaket
4	private	Tidak bisa diterapkan	Tidak dapat dikases dimanapun kecuali oleh method-method yang ada dalam kelas itu sendiri

Aksesabilitas	public	private	protected	default
Dari kelas yang sama	Ya	Ya	Ya	Ya
Dari sembarang kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sembarang kelas di luar paket	Ya	Tidak	Tidak	Tidak
Dari subkelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari subkelas di luar paket	Ya	Tidak	Ya	Tidak

- Polymorism

  - Overloading

    Diperbolehkannya dalam sebuah class memiliki lebih dari satu nama function/method yang sama tetapi memiliki parameter/argument yang berbeda.

```
1  public class Overloading {
2      public void Tampil(){
3          System.out.println("I love Java");
4      }
5      public void Tampil(int i){
6          System.out.println("Method dengan 1 parameter = "+i);
7      }
8      public void Tampil(int i, int j){
9          System.out.println("Method dengan 2 parameter = "+i+" & "+j);
10     }
11     public void Tampil(String str){
12         System.out.println(str);
13     }
14
15     public static void main(String a[]){
16         Overloading objek = new Overloading();
17         objek.Tampil();
18         objek.Tampil(8);
19         objek.Tampil(6,7);
20         objek.Tampil("Hello world");
21     }
22 }
```

- Overriding

  kemampuan dari subclass untuk memodifikasi method dari superclass-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) method superclass-nya. Contoh overriding method dapat dilihat pada subclass “Mobil” yang mendefinisikan kembali method keterangan() dan hapus() dari class “Kendaraan”.